

---

## EXHIBIT C

---

## **Declaration of Nathaniel Adams**

I, Nathaniel Adams, declare pursuant to 28 U.S.C. § 1746, that I have personal knowledge of the following, and if called upon to do so, could and would testify competently to the matters contained herein.

### **1 Qualifications**

I have a Bachelor of Science degree with a major in Computer Science from Wright State University (Dayton, Ohio). I am enrolled in the Graduate School at Wright State University, pursuing a Master of Science degree in Computer Science. I am employed as a Systems Engineer at Forensic Bioinformatic Services, Inc. in Fairborn, Ohio. My duties include analyzing electronic data generated during the course of forensic DNA testing; reviewing case materials; reviewing laboratory protocols; and performing calculations of statistical weights, including custom simulations for casework and research projects. I actively use, develop, and maintain a number of software programs to assist with these efforts. I have been involved in several reviews of probabilistic genotyping analyses in criminal cases, including FST. In 2014 I attended a week-long workshop on interpreting forensic DNA mixtures using emerging software solutions. In January 2016, I was retained in a criminal case and inspected the source code of the commercially-available probabilistic genotyping program STRmix™. Due to a non-disclosure agreement that I signed, I am not allowed to discuss the findings of my review of STRmix™ outside of that particular case.

### **2 Overview**

I have been asked by Sylvie Levine and Christopher Flood to conduct a code review of the Forensic Statistical Tool (FST). I have previously provided a statement dated May 27, 2016 regarding the utility of code reviews when evaluating probabilistic genotyping systems such as FST.

## 3 Setup

### 3.1 Materials provided

For this review I received a set of materials on a flash drive. These materials included a Microsoft Visual Studio solution<sup>1</sup> named “FST\_Production” which includes source code and application configurations. Two database backup files were included in the solution.

I have also received the New York City Office of the Chief Medical Examiner (NYC OCME) “Forensic Statistical Tool Validation” documents in PDF format as well as casework materials describing DNA analyses conducted by OCME in the case US v. Kevin Johnson.

### 3.2 Review environment

Microsoft Visual Studio 2015 Community Edition was used for all code inspection, compilation, debugging, and execution on a PC running Microsoft Windows 7 Professional Service Pack 1 (64-bit). Microsoft SQL Server 2016 Express on a PC running Microsoft Windows 8.1 Professional (64-bit) was used for restoration of the database backups and all database functionality of the FST system. Portions of inspection and testing utilized Microsoft Internet Explorer version 11.0.

### 3.3 Build

In Microsoft Visual Studio, the term “build” is equivalent to the concept of compile-and-go<sup>2</sup>, which automates multiple configurable steps of translating the source code to an executable program, including the compilation<sup>3</sup> step. That equivalency is maintained in this document.

Upon inspection, the majority of the source code appears to be written as a mix of HTML, Javascript, and the C# (pronounced “C sharp”) languages for Microsoft’s ASP.NET (pronounced “A-S-P dot net”) framework. ASP.NET provides a client-server architecture widely used for

---

<sup>1</sup> Solution: “Group of related projects with which a user works.” In this context, a “user” is a user of Visual Studio, e.g. a software developer, rather than a user of the FST software. (Microsoft Developer Network, “Visual Studio 2015 SDK Glossary.” Available: <https://msdn.microsoft.com/en-us/library/bb166253.aspx>)

<sup>2</sup> Compile-and-go: “an operating technique in which there are no stops between the compiling, linking, loading, and execution of a computer program” (ISO/IEC/IEEE, “Systems and software engineering -- Vocabulary,” ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

<sup>3</sup> Compile: “to translate a computer program expressed in a high-order language into its machine language equivalent” (ISO/IEC/IEEE, “Systems and software engineering -- Vocabulary,” ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

distributed, multi-user systems such as websites and applications. FST features a user interface accessible via a web browser running on a local computer. FST also utilizes a user management system for security, a server-side computational engine, and a database for information storage and retrieval, though these systems appear to be hidden from a normal user's interface.

I had to modify portions of code and application configurations in order to get a copy of FST running on my local systems. The steps taken to build a working copy of FST are described in [Appendix A – Refactoring]. Following these steps, I was able to access the FST user interface as described in [Appendix B – FST Interface].

## **4 Assumptions**

### **4.1 Code version**

I assumed that the Visual Studio solution and code contained therein provided to me is identical to the solution used to build the version of FST used in OCME's DNA analysis conducted in the case US v. Johnson, i.e. the "code base" of FST v2.5.

### **4.2 Documentation**

I assumed that materials describing the design and development of FST would be provided to me.

### **4.3 Build and operating environments**

Expected differences between the executables built on my PC and built by OCME are limited to those introduced by different versions of Visual Studio, Microsoft SQL Server, and operating systems used by me and by OCME. I assume these differences are minimal. To explore potential differences, I attempted to replicate the calculations performed by OCME's version of FST used in the case US v. Johnson. Upon comparison, results from my build and OCME's build of FST appear to be indistinguishable for the samples OCME analyzed in the case US v. Johnson. See OCME materials with Bates stamp KJ\_00173 and KJ\_00174 and [Appendix C – US v. Johnson reproductions].

### **4.4 Validation materials**

I assumed that all materials included in the FST validation documents apply to the version of FST provided to me unless indicated otherwise.

## 5 Review

Primary goals of this review included:

Aim 1 - To compare the instructions (source code) and observed behaviors of FST to the requirements, specifications, and design documents (describing intended behaviors) generated during the development and maintenance of FST. A conceptual description of these materials is available in my May 27, 2016 statement.

Aim 2 - To compare the coding style and standards with which FST was developed with practices common to the field of software engineering.

Aim 3 - To perform a limited set of tests on the system to identify any behaviors that detectably depart from intended operations.

Screenshots included in the appendices accompanying this statement are documentation of actions taken during the course of my review.

### 5.1 Documentation

#### 5.1.1 Software engineering materials

As described in an article by Coble, et al. in “DNA Commission of the International Society for Forensic Genetics: Recommendations on the validation of software programs performing biostatistical calculations for forensic genetics applications”<sup>4</sup>, “International industry standards apply to software validation, verification and test documentation.” Standards can also be codified by professional organizations, government agencies, or internal to an organization such as OCME.

In the materials I have been provided, I have not seen any reference to standards governing the development, testing, or validation of FST. Materials contained in FST Validation Vol. 1, “Methods,” could be considered germane to descriptions of program requirements, specification, design, and testing. However, these materials do not cover the entire span of the program such that it could be precisely replicated from these materials or sufficiently tested against during the validation & verification stage of the software development process.

---

<sup>4</sup> M. D. Coble, J. Buckleton, J. M. Butler, T. Egeland, R. Fimmers, P. Gill, L. Gusmão, B. Guttman, M. Krawczak, N. Morling, and others, “DNA Commission of the International Society for Forensic Genetics: Recommendations on the validation of software programs performing biostatistical calculations for forensic genetics applications,” *Forensic Sci. Int. Genet.*, vol. 25, pp. 191–197, 2016.

Enumerated requirements can increase transparency of the development materials by allowing for references made in the source code to the collection of requirements. For example, a comment in the code can state something along the lines of, “satisfies requirement 5.2.2,” could allow a non-author of the source code to connect a particular functional code segment to a particular intended behavior of the software. Ideally, all source code used to build a software program could be attributed to specific requirements defining that program’s intended behavior.

### **5.1.2 Manuals**

A manual for operating FST is included in the FST Validation materials. The provided manual includes instructions and figures that indicate differences between the version of FST described in the validation materials and the version provided to me. See FST Validation Vol. 1, “Methods,” Sec. VI, “Manual,” Figures 1D-E and [Appendix B – FST Interface, Figure B.2 and Figure B.3] accompanying this statement. Notable differences include the “Version 2.5” logo, replacement of a “Degraded Type” dropdown box with a “Lab Kit” dropdown box, and the differences in comparison types available.

From the materials provided to me, it is unclear which portions of the FST software underwent additional development between the time of producing the FST Validation materials and the provision of the FST solution in the case US v. Johnson. In order for the FST Validation materials to provide relevant context to this review (and to casework in general), it is important to fully understand if the differences between versions of FST evaluated during validation and during this review are entirely cosmetic or if any functional behaviors differ.

## **5.2 Testing**

Software testing<sup>5</sup> can be performed in many ways and from many perspectives. The extent of testing expected to be performed during the development of a program is proportional to the importance of its correctness. That is, it is qualitatively more important to conduct extensive testing on a software system regulating a pacemaker than in a video game. While software defects extant in a video game’s code can be potentially overlooked by users or rectified by software updates, injury or loss of life cannot be so simply addressed.

---

<sup>5</sup> Testing: “an activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component” (ISO/IEC/IEEE, “Systems and software engineering -- Vocabulary,” ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

Similarly, the extent of software testing expected of a software developer should be increased when the operation of the software is difficult to test comprehensively prior to its use in a real-world system. Software defects affecting a medical device and a space probe were addressed in my May 27, 2016 statement.

### **5.2.1 Unit tests**

One common perspective of testing is the unit test<sup>6</sup>, whereby individual subcomponents of a larger program are tested for correctness, given pairs of known inputs mapped to known outputs. Allowable inputs and expected behaviors should be defined by the requirements and specifications enumerated for the program.

Subcomponents tested by unit testing can be combined together for integration or component testing. Eventually, the program can be tested in its entirety. As described in my May 27, 2016 statement, there are noted difficulties (or impossibilities) for testing whole probabilistic genotyping systems due to the lack of known ground truths for likelihood ratio calculations performed on noisy casework data.

Some portions of the FST Validation could be considered unit tests, e.g. the tests and results described in FST Validation Vol. 1, "Methods," Sec. IV, "Program Evaluation." However, table 1F in this section describes testing across only 11 of the 15 autosomal loci typed by the Identifiler genotyping kit used in this study, and these 24 total evaluations involved changing multiple variables between tests (numerator and denominator hypotheses as well as template amount).

### **5.2.2 Extent of testing**

Test coverage<sup>7</sup> can be quantified in a number of ways. Branch<sup>8</sup> coverage can be quantified by comparing the number of branches tested to the number of total branches. Path<sup>9</sup> coverage can

---

<sup>6</sup> Unit test: "a test of individual programs or modules in order to ensure that there are no analysis or programming errors." (ISO/IEC/IEEE, "Systems and software engineering -- Vocabulary," ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

<sup>7</sup> Test coverage: "the degree to which a given test or set of tests addresses all specified requirements for a given system or component" (ISO/IEC/IEEE, "Systems and software engineering -- Vocabulary," ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

<sup>8</sup> Branch: "a computer program construct in which one of two or more alternative sets of program statements is selected for execution" (ISO/IEC/IEEE, "Systems and software engineering -- Vocabulary," ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

<sup>9</sup> Path: "in software engineering, a sequence of instructions that may be performed in the execution of a computer program" (ISO/IEC/IEEE, "Systems and software engineering -- Vocabulary," ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

be similarly quantified. Branches or paths of particular importance can be selected for testing or emphasized over less critical parts for more rigorous testing performed during development. Exploration into the amount of testing needed can even help developers reduce a program's complexity before any testing is conducted, proactively reducing the complexity and quantity of tests needed.

It is impossible to objectively test a program's actual behavior against its intended behavior unless its intended behaviors are objectively defined (i.e. in requirements and specifications). Any ambiguities in the specified requirements are necessarily addressed subjectively by the developers. It is difficult to assess the appropriate extent of software testing even when specifications are precisely defined. It is substantially more difficult to determine the appropriate extent of software testing when the object of the testing has no objective standard for correct behavior.

The use of software programs as a component of the *system* of forensic DNA analysis prompts the questions "when" and "how" probabilistic genotyping software systems should be tested. Coble, et al. suggest<sup>4</sup>:

The DNA Commission differentiated developmental from internal (laboratory) validation and emphasizes that the software used is an integral part of the evidential process and **should not be treated as a separate and isolated component.** [emphasis added]

I strongly disagree with this guidance. It is my experience (to the extent that I am free to comment on it) that the development of probabilistic genotyping software is *substantially* less mature than the field of software engineering itself, *especially* in the domain of software testing, validation, and verification. When a probabilistic genotyping system has been developed, it should be tested on its own, in accordance with strictly defined test criteria developed from its own requirements and specifications documents. If it sufficiently passes all tests in isolation, *then and only then* should it be considered for comprehensive testing within the greater forensic DNA analysis process.

If the software is tested in isolation, aberrant behaviors, including outright defects, do not risk being "explained away" by the same noisy profile data that inhibits our ability to assert ground truths for testable likelihood ratios. The field of software engineering has had its own standards for validating software for decades, and it is very reasonable to draw on these general standards directly.



For reference, the Scientific Working Group on Forensic DNA Analysis Methods published a 12-page document<sup>10</sup> in 2015 on validating probabilistic genotyping systems, only a portion which applied to the development of software, and *none* of which addressed software engineering standards. The International Society for Forensic Genetics (ISFG) published a 15-page document<sup>4</sup> in 2016 on validating probabilistic genotyping systems. ISFG makes reference to two long-extant, 100+ page documents<sup>11,12</sup> published by the Institute of Electrical and Electronics Engineers (both of which are already marked as superseded) *and* a 14-year old guidance document<sup>13</sup> from the Food and Drug Administration. Multiple frameworks for validating general-purpose software have long been codified and continue to be revised by organizations such as IEEE. A framework for validating biological software with complex and safety-risk-prone components has been considered and codified by the federal government. While the field of forensic DNA has no public and widely used software standards specific to the field, we can draw on standards describing good practices that generally apply to all software development processes.

I have seen no quantitative description of test coverage in the FST Validation materials. The time that it would take to quantitatively evaluate the extent of testing FST has undergone is outside the scope of this review.

### 5.2.3 Test code and automated software testing

“Test” code can be written parallel to a software program’s code and is one of the most common practices throughout software engineering. The purpose of test code is not to provide any functional behavior or fulfill any requirement of the software, but rather to ensure that the program’s code fulfills its requirements by testing its actual behavior against its expected behavior. Unit tests are commonly written as code segments within the software’s solution. Multiple test frameworks exist for assisting developers with automated testing, some even writing the code automatically. Modern development environments, including Visual Studio,

---

<sup>10</sup> Scientific Working Group on DNA Analysis Methods. Guidelines for the Validation of Probabilistic Genotyping Systems. Available at - <http://www.swgdam.org/publications>

<sup>11</sup> IEEE Standard for Software and System Test Documentation. (2008). IEEE Std 829-2008.

<sup>12</sup> IEEE Standard for System and Software Verification and Validation . (2012). IEEE Std 1012-2012 (Revision of IEEE Std 1012-2004).

<sup>13</sup> Food and Drug Administration. (2002). General principles of software validation; final guidance for industry and FDA staff. Available at - <http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm085281.htm>

include features for executing automated software tests each time the program is built, ensuring that no new defects are detected after modifications have been made to the program's code.

A notable advantage of writing test code is that, even in the absence of precise requirements and specifications definitions, test code can serve as a de facto specification since all tests should have explicit pass/fail criteria. These tests' pass/fail criteria can implicitly describe a program's expected behaviors. An entire approach to software development, called test-driven development, utilizes this approach to software development rather than the traditional hierarchical process described in the appendix of my May 27, 2016 statement.

Similar to unit tests, higher-level component and integration testing, regression testing, and random testing can be automatically executed prior to building a solution or as a step of the compile-and-go build process. A practice called "nightly builds" involves taking the snapshot of the current code in a version control system and attempting to build the solution, triggering any automated testing processes specified therein. In this way, developers are more prone to notice software defects caused by their recent modifications, allowing them to make any needed changes before the defect becomes more tightly integrated into the system.

I have seen no code indicating that any test code has been written for, or automated software testing has been performed on, FST.

### **5.3 Validation and verification (v&v)**

A definition of validation is provided by IEEE as, "The confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application are fulfilled." It is necessarily difficult to validate a program for which explicit requirements and specifications have not been established.

It is worth considering, due to the incomplete and at-times ambiguous descriptions of FST's intended behaviors, whether FST has been validated, partly validated, or not yet validated from a software engineering perspective.

Using IEEE's definition of verification, "The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase," it is difficult to assess what verification processes have been undertaken during FST's validation and continued use.

I have seen no document resembling a software development plan describing stages of development, against which comparisons for verification of FST could be made.

It is reasonable to expect that unanticipated requirements changes were made following the failure to validate FST for four-person mixtures, which presumably gave rise to the note on the login screen seen in [Appendix B– FST Interface, Figure B.1]. Since this change necessarily followed at least a portion of the validation study conducted by OCME, and along with the changes and undocumented behavior noted in [5.1.2 Manuals and 5.6 Undocumented behavior], further investigation is indicated into the software validation and verification processes conducted on FST.

### 5.3.1 Perception of v&v

Overlapping terms of art between software engineering and forensic biology should be distinguished. A “validation study” in the field of forensic DNA is ultimately intended to determine the limits of certainty with which a laboratory, utilizing certain processes, can confidently arrive at a given conclusion. Ideally, the confidence with which the conclusion is stated and the significance of the conclusion are quantified. The manifestation of forensic DNA’s “statistical weights” in the form of random match probabilities, probabilities of inclusion, and likelihood ratios are direct responses to this ideal.

However, validation of a software system is a comparison of a program’s intended and actual behaviors. The limits of confidence with which the software can report a particular conclusion is part of a greater process of scientific inquiry and is *not* part of the validation efforts from a software engineering perspective. Conflation between these two concepts is detrimental to understanding the relationship of both. A *validation study* of a forensic DNA analysis process involving software should incorporate the uncertainties inherent to the software system(s) and should only follow the conclusion that the software itself has been validated.

### 5.3.2 Independence

Regarding the independence of software validation, the FDA states<sup>13</sup>:

Validation activities should be conducted using the basic quality assurance precept of "independence of review." Self-validation is extremely difficult. When possible, an independent evaluation is always better, especially for higher risk applications.

IEEE defines<sup>22</sup> “independent” as, “performed by an organization free from control by the supplier, developer, operator, or maintainer.”

I am unaware of any independent review of FST's development process prior to this review.

## **5.4 Code maintenance**

### **5.4.1 Code ownership**

The concept of code ownership is that the developer who originally wrote or currently maintains the code is an authority on its behavior. Should a future developer uncover issues in the existing code or create issues when modifying the code, the author can be used as a human reference. Over 100 source code files are included in the FST solution, each presumably having been authored or at least modified by a human developer. I conducted a keyword search of the entire solution for terms indicating authorship of original code or modifications made to existing code. I located three files that have comments<sup>14</sup> explicitly describing authorship. I located only five files that had any comments attributing a contribution or modification to a particular developer. Developers noted in these comments are Vivien Song, Dhrubajyoti "Dhruba" Chattopadhyay, and "win." I found no other references to whomever "win" is.

Overall, it is unclear from my review who authored the majority of the code constituting the FST solution. It is possible that inspection of the version control system could provide further information. See [5.4.4 Version control] for further discussion.

### **5.4.2 Code attribution**

Numerous references to the popular coding website Stack Overflow<sup>15</sup> are observed throughout the code, including the sole comment describing the routine "ByteArrayHelper" in the file "IndividualReportPrinter.cs". Without any elaboration as to the underlying function of the routine, the comment simply states, "/// This helper class is used to find patterns in byte arrays. Found on stack overflow. Thank you, Internet!"

Taking code from unknown sources can lead to using code that is poorly understood and possibly dangerous. No URL's to the original Stack Overflow web page are provided for reference by future developers. The original author is not acknowledged, nor is contact information for him/her provided. Further implications of using unattributed, unlicensed, "borrowed" code is outside of the scope of this review.

---

<sup>14</sup> Comment: "information embedded within a computer program, job control statements, or a set of data that provides clarification to human readers but does not affect machine interpretation" (ISO/IEC/IEEE, "Systems and software engineering -- Vocabulary," ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

<sup>15</sup> <http://stackoverflow.com/>

### 5.4.3 Coding style

#### 5.4.3.1 Coding conventions

The Microsoft Developer Network describes coding style guidelines<sup>16</sup>:

The C# Language Specification does not define a coding standard. However, the guidelines in this topic are used by Microsoft to develop samples and documentation.

Coding conventions serve the following purposes:

- They create a consistent look to the code, so that readers can focus on content, not layout.
- They enable readers to understand the code more quickly by making assumptions based on previous experience.
- They facilitate copying, changing, and maintaining the code.
- They demonstrate C# best practices.

Coding style guidelines are generally stylistic, rather than functional, conventions governing the development of software. A Google search for the phrase “c# guide style OR conventions” returned 1.83 million search results on October 26, 2016. Top results include coding style guides or conventions published by Microsoft, academic institutions, private individuals, and companies. There is no one “correct” style guide, but consistent stylistic practices throughout an organization, or at least a specific solution, increase code comprehension and consequently decrease the rate and significance of software defects<sup>17</sup>.

I found no references to coding conventions or style guides in the FST solution or FST Validation materials. Upon inspection of the FST source code, disparate coding styles are indicated. While portions of some code feature inline comments for nearly every line of code, long sections of code are uncommented in other files, with no annotations including even summary descriptions of purpose. Without explicit descriptions of intended behaviors, reviewers and developers are left to infer the purpose of the code from its own function. In the absence of

---

<sup>16</sup> “C# Coding Conventions (C# Programming Guide)” (Microsoft Developer Network. Available at - <https://msdn.microsoft.com/en-us/library/ff926074.aspx>)

<sup>17</sup> Defect: “a problem which, if not corrected, could cause an application to either fail or to produce incorrect results” (ISO/IEC/IEEE, “Systems and software engineering -- Vocabulary,” ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

external definitions of intended functionality, a developer runs the risk of justifying the behavior of existing code by its sheer existence. See also [5.2 Testing].

#### 5.4.3.2 Code smells

Martin Fowler describes code smells as, “A code smell is a surface indication that usually corresponds to a deeper problem in the system.”<sup>18</sup> In this sense, a *smell* is not a defect in itself but is a deviation from good coding practices, which can indicate underlying software defects. Coding conventions and style guides purposely prevent code smells. Coding practices such as writing long routines<sup>19</sup> or complex classes<sup>20</sup> can obfuscate underlying issues due to the difficulty of comprehending large segments of code.

The FST source code presents a number of basic smells such as routine length and complex classes. For example, the “DoCompare” routine involved in computing likelihood ratios is approximately 200 lines long, including comments and whitespace. The “configureComparison” method differentiates between 38 different hypothesis pairs and constitutes approximately 400 lines of the “ComparisonData.cs” file that is, itself, over 800 lines long. Routines such as Database.SaveCase require 20 or more ordered input parameters. An uncertain developer invoking this routine could potentially transpose two of these input parameters, resulting in aberrant behavior that might go undetected.

Some source code files consisting of only a single class’ implementation are exceptionally long. The “Database.cs” class file is approximately 1,900 lines long and defines only one class – “Database”.

Magic numbers<sup>21</sup> are used throughout the FST solution. In the event of a needed change to these values during development, all hard-coded values must be located and modified since

---

<sup>18</sup> Fowler, M. “CodeSmell”. Available at - <http://martinfowler.com/bliki/CodeSmell.html>

<sup>19</sup> Routine: “a subprogram that is called by other programs and subprograms” (ISO/IEC/IEEE, “Systems and software engineering -- Vocabulary,” ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

<sup>20</sup> Class: “1. an abstraction of the knowledge and behavior of a set of similar things. 2. a static programming entity in an object-oriented program that contains a combination of functionality and data. Syn: type NOTE Classes are used to represent the notion of ‘things whose knowledge or actions are relevant’.” (ISO/IEC/IEEE, “Systems and software engineering -- Vocabulary,” ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

<sup>21</sup> Magic number: “literal value that is used by a program directly rather than being embedded in a named constant or variable” (ISO/IEC/IEEE, “Systems and software engineering -- Vocabulary,” ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

they are dispersed throughout the code rather than being centrally, or at least consistently, located.

More subjective code smells are prevalent in FST, such as excessive commenting of code [see 5.4.3.3 Comments] and long variable names such as the 131-character-long “compareMethodIDSerializationBackupDoNotMessWithThisVariableBecauseItsAnUglyHackAroundACrappyLimitationOfTheFCLsCrappyXMLSerializer” seen in the file “ComparisonData.cs”.

### 5.4.3.3 Comments

Comments provide annotations for developers to reference but do not affect the functional behavior of the source code. Comments can be generated manually by developers or automatically by a development environment (e.g. Visual Studio). In the C# language, lines preceded by a double-slash (“//”) are not translated to machine code and can therefore serve as non-functional annotations of the code. Similarly, lines of old or deprecated code can be prepended with a “//” to remove from the functional body of code – this is often observed during the development stage of a software program.

Many comments in the source code I have reviewed appear to be manually generated by developers, e.g. line 353 of “Comparison.cs”, which states “// here we iterate to find the LR for each of the races (they are different because the frequency values are different per-race)”. Such comments serve to increase comprehension of the code and are ultimately beneficial to the software development efforts. Elsewhere, many of the comments appear to be auto-generated by Visual Studio without being sufficiently filled-in with English-language descriptions of program behavior. These differences result in an inconsistent commenting convention.

However, many portions of the FST code are without comments for dozens or hundreds of lines, including some entire routines and class definitions. As stated above, the same mechanism that indicates to the compiler that a line is not to be considered code (i.e. a literal “comment”) can be used to remove lines of actual code from the functional behavior of a program. In the file “ManageUsers.aspx.cs”, there are approximately 150 lines of uncommented code with four lines containing code that are prepended by “//”.

Other large sections of code are entirely “commented-out,” indicating a change in requirements or implementation of that section of code. Reasons for maintaining “commented-out” code rather than deleting it are future reference, future re-implementation, use of the code as a test during development, or simply untidy maintenance practices.



Excessive commenting can detract from developer comprehension of the code. In the same file described above, “ComparisonData.cs”, a sequence of otherwise intuitively-named variables is interspersed with comments, needlessly breaking up the flow:

```

66      // add one comparison to the ID list
67      NumeratorProfiles.ComparisonIDs.Add(1);
68      // add one unknown in the denominator
69      DenominatorProfiles.UnknownCount = 1;
70      // set the long name
71      CompareMethodLongName = "Comparison / Unknown";
72      // set the short name
73      CompareMethodShortName = "C / U";
74      // set up the numerator header format string
75      hpHeadFormatString = "{0} (Comparison)";
76      // set up the denominator header format string
77      hdHeadFormatString = "Unknown";

```

Figure 1 – ComparisonData.cs lines 66-77.

Excessive comments detract from the readability of the code, in this case, doubling the length of the code segment. Excessive comments increase maintenance time costs and complexity, as code modified during the normal development cycle or as a bug fix must be accompanied with any needed changes to the comments describing the code’s intended (modified) behavior.

#### 5.4.4 Version control

IEEE defines “version control” as, “establishment and maintenance of baselines and the identification and control of changes to baselines that make it possible to return to the previous baseline”<sup>22</sup>. Normally, when a file is saved to a hard drive, the previous version of the file is overwritten by the new version, permanently discarding any previous information contained therein. Version control systems allow for iterative changes to be made to all files in the system, which in turn emphasizes code ownership and allows for modifications to be tracked.

Common technologies freely providing version control functionality include Apache’s Subversion (SVN) and Git. I found references to SVN in only the FST.Web submodule’s “FST\_Production\_Service” folder, indicating that the use of SVN version control might have only applied to a portion of the FST system, specifically its report-generating feature. I observed no references to Git version control systems or any other version control system.

<sup>22</sup> ISO/IEC/IEEE, “Systems and software engineering -- Vocabulary,” ISO/IEC/IEEE 24765:2010(E), vol. 2010.



From the materials provided to me, it is unclear if the FST code base is maintained in a version control system.

#### **5.4.5 Dead code**

As described by Debray, et al<sup>23</sup>, “dead code refers to computations whose results are never used.” Visual Studio provides basic detection of dead code upon attempting to build a solution. Dead code identified in the FST solution by Visual Studio include a number of import “directives” (i.e. imported packages<sup>24</sup>) specified in many files that are never used in those files. Possible causes for unused imports are modified specifications or designs; updates to the ASP.NET framework between versions of Visual Studio; and import code copied and pasted between files with different package requirements. From the materials provided to me, I cannot be certain as to why these unused imports are included in the FST solution.

### **5.5 FST Database**

#### **5.5.1 Empty tables**

The “FST\_DB” database backup I was provided includes tables named “Cases,” “Known,” and “Tests.” These tables are empty, unlike other tables in this database, which contain information on dropout rates, allele frequencies, etc. I do not know if entries in these tables exist in OCME’s version of the database but were removed before the database backup was made for provision in this case; if I was provided a “test” or “developmental” database backup; or if these tables are also empty in the FST build used by OCME for casework.

#### **5.5.2 “CaseTypes” table and “ComparisonData.cs”**

The same “FST\_DB” database backup I was provided includes a table name “CaseTypes.” It contains six entries:

---

<sup>23</sup> Debray, S. K., Evans, W., Muth, R., & De Sutter, B. (2000). Compiler techniques for code compaction. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 22(2), 378–415.

<sup>24</sup> Package: “a separately compilable software component consisting of related data types, data objects, and subprograms” (ISO/IEC/IEEE, “Systems and software engineering -- Vocabulary,” ISO/IEC/IEEE 24765:2010(E), vol. 2010.)

```

SELECT TOP (1000) [Case_Id]
, [Case_Desc]
, [Status]
, [Ordinal]
FROM [FST].[dbo].[CaseTypes]

```

100 %

Results Messages

	Case_Id	Case_Desc	Status	Ordinal
1	1	Comparison/Unknown	1	1
2	2	Comparison + Known / Known + Unknown	1	2
3	3	Comparison + Unknown / 2 Unknowns	1	3
4	4	Comparison + 2 Unknowns / 3 Unknowns	1	4
5	5	Comparison + Known + Unknown / Known + 2 Unknowns	1	5
6	6	Comparison + 2 Knowns / 2 Known + Unknown	1	6

*Figure 2 - "CaseType" database table contents*

The contents of the table "CaseTypes" appear to be pairs of hypotheses representing numerators and denominators for likelihood ratio calculations. The source code file named "ComparisonData.cs," within routine named "configureComparison," describes 38 similar-looking hypotheses:

```

65         case 1:
66             // add one comparison to the ID list
67             NumeratorProfiles.ComparisonIDs.Add(1);
68             // add one unknown in the denominator
69             DenominatorProfiles.UnknownCount = 1;
70             // set the long name
71             CompareMethodLongName = "Comparison / Unknown";
72             // set the short name
73             CompareMethodShortName = "C / U";
74             // set up the numerator header format string
75             hpHeadFormatString = "{0} (Comparison)";
76             // set up the denominator header format string
77             hdHeadFormatString = "Unknown";
78             break;
79         case 2:
80             NumeratorProfiles.ComparisonIDs.Add(1);
81             // set a known in the numerator
82             NumeratorProfiles.KnownIDs.Add(1);
83             // set the same known in the denominator
84             DenominatorProfiles.KnownIDs.Add(1);
85             DenominatorProfiles.UnknownCount = 1;
86             CompareMethodLongName = "Comparison + Known / Known + Unknown";
87             CompareMethodShortName = "C + K / K + U";
88             // we use {4} to denote the known name
89             hpHeadFormatString = "{0} (Comparison) + {4} (Known)";
90             hdHeadFormatString = "{4} (Known) + Unknown";
91             break;

```

Figure 3 - Cases 1 and 2 from "ComparisonData.cs", routine "configureComparison"

From my inspection of the behavior of the FST system, the user interface shown in [Appendix B— FST Interface, Figure B.3] displays six possible comparisons which correspond to the "CaseTypes" table entries. It is my understanding that the comparisons performed by OCME in the case US v. Johnson are of types #3, "Comparison + Unknown / 2 Unknowns," and #4, "Comparison + 2 Unknowns / 3 Unknowns." See OCME materials with Bates stamp KJ\_00173 and KJ\_00174.

Attempts to select the third and fourth items from the comparison dropdown list resulted in the error page seen in [Appendix B – FST Interface, Figure B.4]. It seemed that while items #3 and #4 in this dropdown list describe the comparisons made by the OCME in the case US v. Johnson, the FST version I built was incapable of performing these comparisons as described. The dropdown list appears to pull its contents from the "CaseTypes" table in the database rather than from the source code of the program, itself.

However, upon further inspection, it appears that the comparisons dropdown list is merely a placeholder for parallel functionality in the code, e.g. the first dropdown list item maps to the calculation described under "case 1" in [Figure 3] above. I inspected the code to find:

```

92         case 3:
93             // this comparison ID is unused at this time
94             throw new NotImplementedException();
95         case 4:
96             // this comparison ID is unused at this time
97             throw new NotImplementedException();
98         case 5:
99             NumeratorProfiles.ComparisonIDs.Add(1);
100            NumeratorProfiles.UnknownCount = 1;
101            DenominatorProfiles.UnknownCount = 2;
102            CompareMethodLongName = "Comparison + Unknown / 2 Unknowns";
103            CompareMethodShortName = "C + U / 2 U";
104            hpHeadFormatString = "{0} (Comparison) + Unknown";
105            hdHeadFormatString = "2 Unknowns";
106            break;
107         case 6:
108            NumeratorProfiles.ComparisonIDs.Add(1);
109            NumeratorProfiles.UnknownCount = 2;
110            DenominatorProfiles.UnknownCount = 3;
111            CompareMethodLongName = "Comparison + 2 Unknowns / 3 Unknowns";
112            CompareMethodShortName = "C + 2 U / 3 U";
113            hpHeadFormatString = "{0} (Comparison) + 2 Unknowns";
114            hdHeadFormatString = "3 Unknowns";
115            break;

```

*Figure 4 - Cases 3-6 from "ComparisonData.cs", routine "configureComparison"*

The reason selecting #3 or #4 from the list resulted in an error page is because "case 3" and "case 4" trigger exceptions in the code. When I selected options #5 or #6 from the dropdown list, the comparisons performed appear to not be the comparisons listed in the dropdown box (corresponding to the "CaseTypes" database table), but rather the comparisons described in "case 5" and "case 6," respectively, in the "configureComparison" routine as seen in [Figure 4] above. That is, I had to select the comparison listed as "Comparison + Known + Unknown / Known + 2 Unknowns" in order to perform the actual comparison "Comparison + Unknown / 2 Unknowns." Additionally, I had to select the comparison listed as "Comparison + 2 Knowns / 2 Knowns + Unknown" in order to perform the actual comparison "Comparison + 2 Unknowns / 3 Unknowns."

The comparisons described in [Appendix C – US v. Johnson reproductions] appear to support this inference, including an artifact of the database table "CaseTypes" descriptions of the comparisons appearing in the "Debug out" files shown in [Appendix C– US v. Johnson reproductions, Figure C.5 and Figure C.10].

It is unclear to me why this aberrant behavior occurs or if this behavior of FST is also observed by FST users at OCME during casework analysis.

## 5.6 Undocumented behavior

An instance of undocumented behavior of the FST program is noted in the calculation of likelihood ratios performed in "Comparison.cs". A routine included in the source code file ("Comparison" class), named "CheckFrequencyForRemoval" appears to perform the following behavior:

1. Check all replicate (evidentiary) genotypes for any locus that contains alleles whose frequency sums to  $\geq 0.97$  in any of the four subpopulations ("Asian," "Black," "Caucasian," and "Hispanic").
2. Remove these loci from the likelihood ratio calculations.

During this review, I encountered no notice, either intended or actual, provided to the user of FST that any loci were removed from the likelihood ratio calculation. I found no indication that this behavior is intended during my examination of FST-related publications and the FST Validation materials. Testing I performed [see Appendix D – Single-locus Testing] indicates that this behavior is real. Equivalent locus likelihood ratios of "1" were reported for single-locus tests, regardless of whether the "comparison" (reference) profile shared alleles with the evidentiary item or not, as long as the sum of allele frequencies was  $\geq 0.97$  at that locus.

It is unclear from this review how often this is encountered during casework; whether an FST user is aware that a locus is ignored when it is ignored by the FST software; or how this has affected or can affect casework mixture analysis and reporting. I am not aware of any studies conducted on this feature's impact on casework samples and the calculation of their statistical weights reported by OCME.

Such departures from the published descriptions of FST's behavior during its actual operation raise the question if additional undocumented behaviors, either intended or actual, affect casework samples during the operation of the FST software or the reporting of its likelihood ratio results.

## 6 Conclusion

This review should not be considered complete, comprehensive, or exhaustive. I am available to continue reviewing the materials I have been provided or to review additional materials previously not provided.

I am ultimately unsure if the materials provided to me are the same materials that were used while conducting the study documented in the FST Validation materials or casework analyses in the case US v. Johnson.

Reviewing the materials I was provided, I did not leave with the impression that FST was developed by an experienced software development team, especially in regards to adherence to coding conventions, use of general software development standards, or even basic good practices such as using consistent coding styles; attributing authorship to code segments; or writing automated software tests.

The presence of undocumented behaviors and differences in appearance between the version I reviewed and the version shown in the FST Validation materials calls into question when development on FST ceased and how this affects or might affect the significance of any scientific conclusions predicated on FST Validation materials.

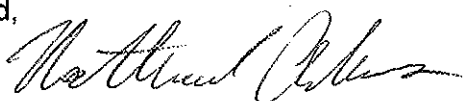
Given the lack of a set of detailed and explicit software requirements and specifications, it is difficult to evaluate to what extent FST's testing reflects rigorous and appropriate software testing. Consequently, the idea that FST is a "validated" software program should be revisited and a more thorough review conducted.

Given the materials I was provided, the time I was able to review them, and the issues I identified, I recommend that:

1. Explicit requirements and specifications of the intended (not necessarily extant) behaviors be written such that no undocumented or unexpected behaviors exist and against which software testing of FST can be conducted.
2. A formal software test plan be written and described in terms of coverage.
3. Software tests needed for validation be written as automated tests that can be executed and reported on by an automated software testing framework.
4. The FST Validation materials and publications regarding FST be compared to the validated version of FST, with any discrepancies noted.

Until these steps are followed, the correctness of the behavior of the FST software should be seriously questioned.

Signed,



October 27, 2016

## **Appendix A – Refactoring**

I made modifications to the database, application configurations, and code in order to assemble a working FST client-server system on my local PC's.

### **A.1 Databases**

Aside from the modifications described below, I made no further changes to the FST databases I was provided.

#### **A.1.1 Restoration of backups**

I received backup copies of two databases, named "FST\_DB.bak" and "FST\_Membership.bak." I restored these databases into Microsoft SQL Server.

#### **A.1.2 Database security**

The restored databases contained user profiles and logins with security identifiers (SIDs) that I was not provided. I was able to insert a new user into each database with full privileges to bypass the need for transferring or replicating existing SIDs.

### **A.2 Application configurations**

Aside from the modifications described below, I made no further changes to FST's application configuration.

#### **A.2.1 .NET packages**

Multiple packages freely provided by Microsoft are required for building FST:

- Microsoft.ReportViewer.Common
- Microsoft.ReportViewer.WebForms
- Microsoft.ReportViewer.WinForms

I received these packages on the flash drive containing the FST solution but I had to reestablish references from the solution to the packages prior to building the solution.

#### **A.2.2 Database connections**

The FST solution has multiple references to a database server. These references are server-specific, so they had to be regenerated by my server and specified in multiple configuration files to point to my local server.



### **A.2.3 Project properties**

I changed the “Startup Project” property to attempt to start all four major components of the FST solution: FST.Common, FST.Web, FST\_WindowsService, and FSTServiceConsoleHost. FST\_WindowsService had to be manually installed and started to operate.

### **A.2.4 FST\_WindowsService**

Using the Visual Studio developer’s command prompt, I used the Microsoft-provided ‘installutil.exe’ program to install the FST\_WindowsService “debug”-built executable as a background Windows service. I then started the service manually.

The service is named “FST\_Charles”.

## **A.3 Code modifications**

Aside from the modifications described below, I made no changes to the FST solution’s source code.

### **A.3.1 Secure login**

When a user first connects to the FST server (via a web browser), the user is presented with a login page that requires a username and password. The login attempt is negotiated by the project FST.Web’s “Login.aspx.cs” file, specifically the routines “btnLogin\_Click” and “IsAuthenticated.” I commented-out portions of this code and set the IsAuthenticated method to return “true” regardless of the login credentials supplied.

By this method, I am able to “log in” using the “admin” login without providing a legitimate password since no password check is performed.

## Appendix B – FST Interface

Forensic Statistical Tool v2.5

Username:

Password:

Login

The Forensic Statistical Tool has been validated to be used on the following sample situations:  
Samples amplified with more than 100pg of total DNA with Identifier 28 cycles  
Samples amplified with less than 100pg of total DNA with Identifier 31 cycles  
Single source samples, two-person mixtures in which one or more components were not deconvoluted, or three-person mixtures in which two or more components were not deconvoluted  
1 - 3 replicates of different amplifications  
Calculations can be done on a sample that is deemed to be "deducible" or "non-deducible" and can be calculated with up to two assumed contributors (or "knowns"). The number of contributors must be the same for the numerator and denominator.

Any other set of scenarios was **not validated** and therefore **will not give reliable results**. Please contact a supervisor or Technical Leader before proceeding with this analysis if you have any questions. By logging in, you acknowledge these specific parameters.

© 2016 Office Of Chief Medical Examiner. The City Of New York. All Rights Reserved.

Figure B.1 – Login screen for FST.

The screenshot shows a web browser window with the address bar displaying `http://localhost:2926/frmDefault.aspx` and the title "Forensic Statistical Tool". The browser's menu bar includes "File", "Edit", "View", "Favorites", "Tools", and "Help". The application header features a blue navigation bar with "Home" and "Admin" links, and a user status area showing "Welcome admin" and a "Logout" link. The main content area is titled "Scenarios:" and contains a form with the following elements:

- A "Select Scenario:" dropdown menu currently showing "Comparison/Unknown".
- A "Lab Kit:" dropdown menu currently showing "Identifier".
- A parameter  $\theta$  set to 0.03.
- A section titled "Comparison Profile 1:" with a text input field for "Comparison 1 Name".
- Two blue buttons at the bottom right: "Bulk" and "Go".

Figure B.2 – Homepage for new analysis.

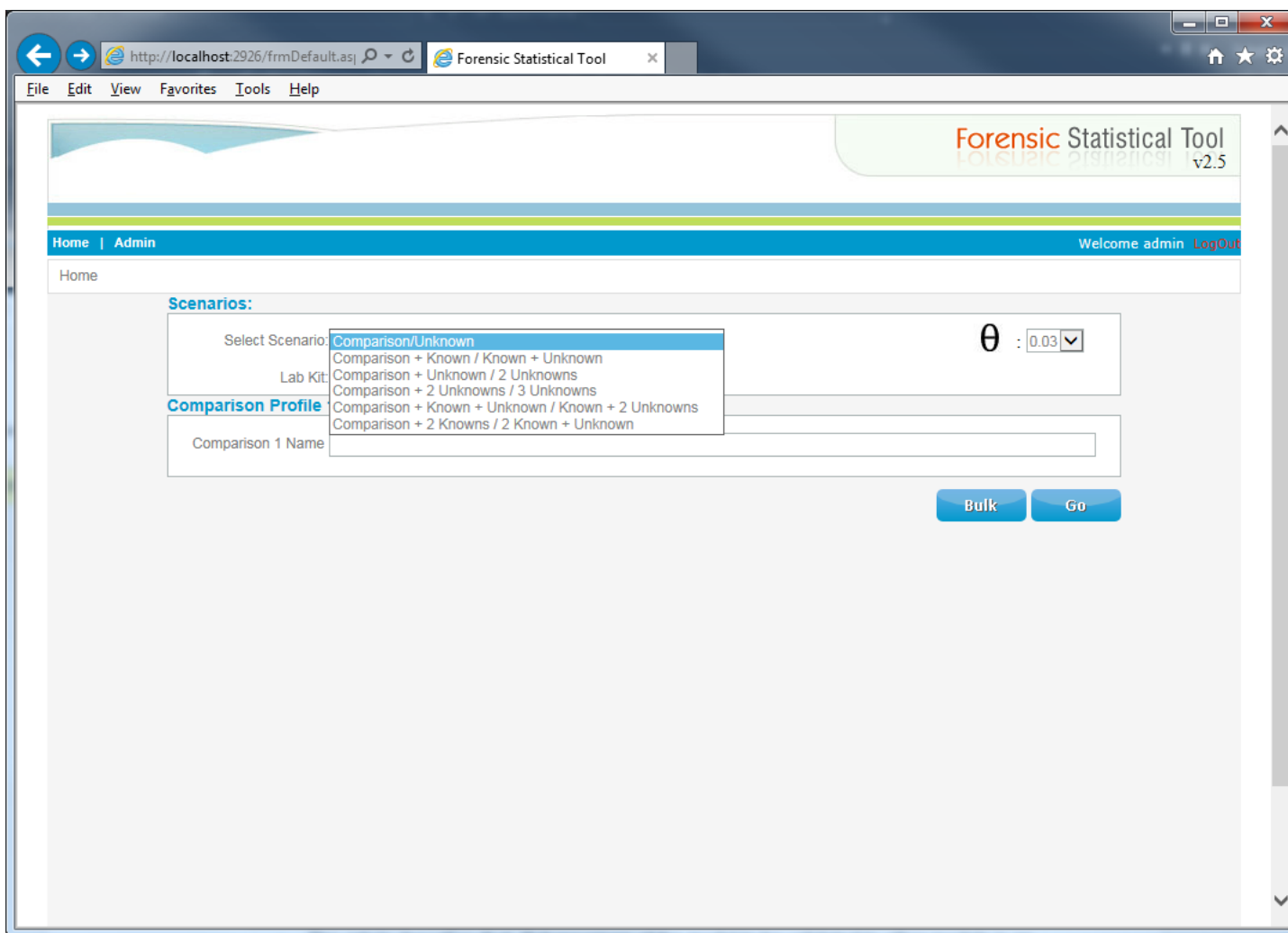


Figure B.3 – Available “case types” dropdown menu on homepage for new analysis.

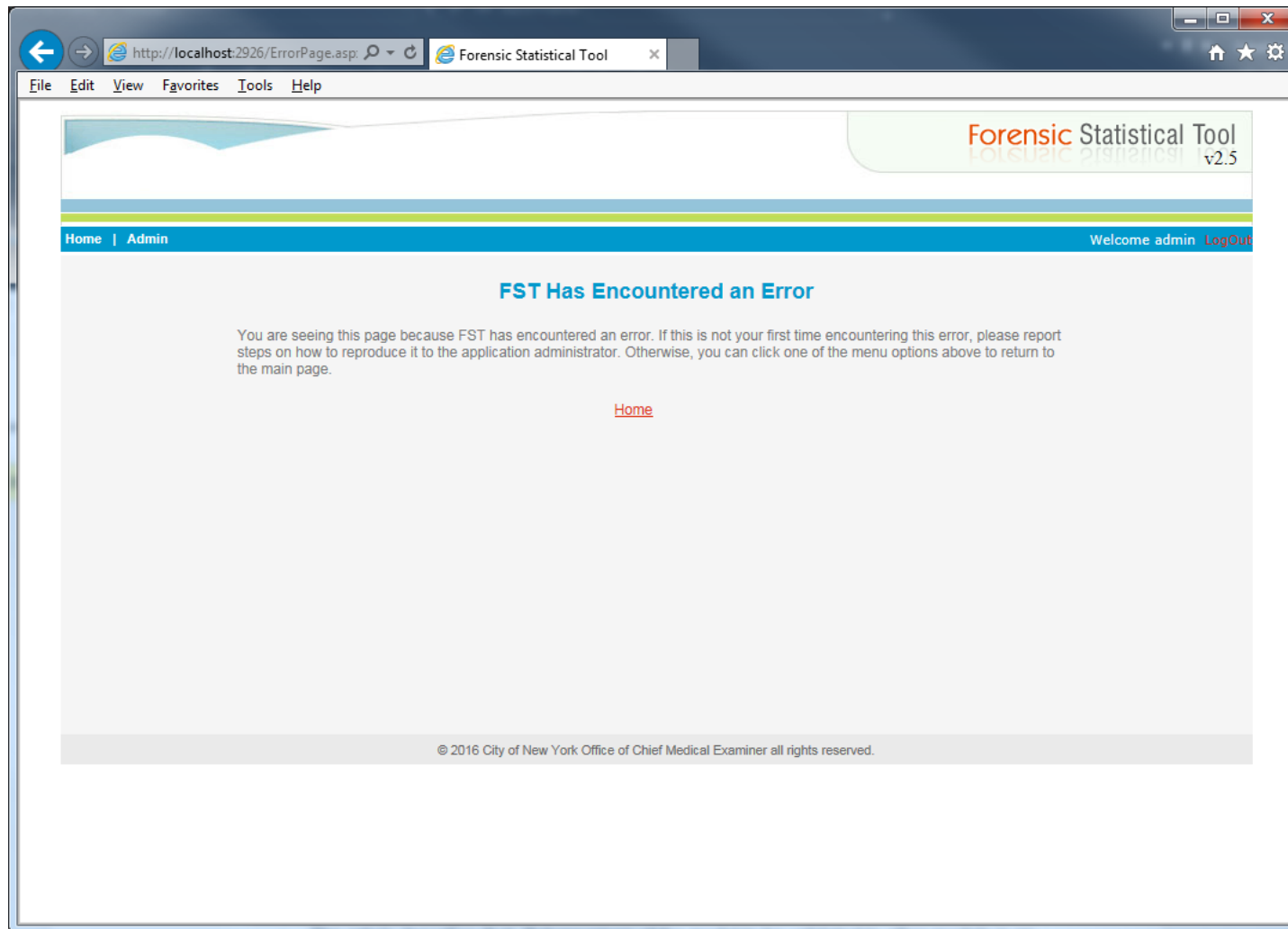


Figure B.4 - FST error page.

## Appendix C – US v. Johnson reproductions

The screenshot displays the 'Forensic Statistical Tool v2.5' web application. The interface is divided into several sections:

- Header:** Includes a navigation bar with 'Home' and 'Admin' links, and a 'Welcome admin' message with a 'Logout' link.
- Breadcrumb:** Shows the current path: 'Comparison + 2 Unknowns / 3 Unknowns' > 'Comparison + 2 Unknowns / 3 Unknowns (Edit)'.
- Form Fields:**
  - FB#1:** 'FST report replicate'
  - Comparison:** 'Kevin Johnson test'
  - FB#2:** (Empty)
  - Item:** 'back strap and grips # 2'
- Evidence Section:**
  - Three tabs: 'Replicate 1' (selected), 'Replicate 2', and 'Replicate 3'.
  - A grid of DNA profile data for Replicate 1:
 

Marker	Value
D8S1179	17
D21S11	31
D7S820	12
CSF1PO	12
D3S1358	17
TH01	9
D13S317	12
D16S539	13
D2S1338	21
D19S433	15.2
vWA	16
TPOX	8
D18S51	15
D5S818	13
FGA	INC
- Kevin Johnson test (Comparison):**
  - Deducible:** 'No' (selected)
  - DNA Template Amount (pg):** '175'
  - Buttons:** 'Compare', 'Reset', 'Back'
- Footer:** '© 2016 City of New York Office of Chief Medical Examiner all rights reserved.'

Figure C.1 - Replicate 1 for "back strap and grips # 2"

Forensic Statistical Tool v2.5

Home | Admin Welcome admin [LogOut](#)

Comparison + 2 Unknowns / 3 Unknowns Comparison + 2 Unknowns / 3 Unknowns (Edit)

FB#1:  Comparison:  FB#2:  Item:

**Evidence:**

Replicate 1 **Replicate 2** Replicate 3

D8S1179 17 <input type="text"/> 13,14,15,16,17	D21S11 31 <input type="text"/> 27,28,30,31	D7S820 12 <input type="text"/> 8,12	CSF1PO 12 <input type="text"/> 12	D3S1358 16 <input type="text"/> 14,15,16
TH01 9 <input type="text"/> 6,7,9	D13S317 12 <input type="text"/> 11,12	D16S539 13 <input type="text"/> 11,12,13	D2S1338 19 <input type="text"/> 18,19	D19S433 16.2 <input type="text"/> 11,13,14,15,15.2,1
vWA 17 <input type="text"/> 16,17	TPOX 9 <input type="text"/> 8,9	D18S51 21 <input type="text"/> 18,21	D5S818 13 <input type="text"/> 11,13	FGA 22 <input type="text"/> 22

**Kevin Johnson test (Comparison):**

Deducible:  DNA Template Amount (pg):

© 2016 City of New York Office of Chief Medical Examiner all rights reserved.

Figure C.2 - Replicate 2 for "back strap and grips # 2"

The screenshot shows a web browser window with the URL `http://localhost:2926/frmManualEn` and a tab titled "FST - Comparison + 2 Unkn...". The application is titled "Forensic Statistical Tool v2.5". The navigation bar includes "Home" and "Admin", with a welcome message "Welcome admin" and a "LogOut" link. The breadcrumb trail is "Comparison + 2 Unknowns / 3 Unknowns" > "Comparison + 2 Unknowns / 3 Unknowns (Edit)".

The main form contains the following fields:

- FB#1:** FST report replicate
- Comparison:** Kevin Johnson test
- FB#2:** (empty)
- Item:** back strap and grips # 2

Below these fields is a section titled "Evidence:" with a sub-section "Kevin Johnson test (Comparison):". This section contains a grid of 15 DNA marker comparison boxes, each with a label, a dropdown menu, and a text input field:

D8S1179 16 ▼ 16 ▼ 16,16	D21S11 27 ▼ 30 ▼ 27,30	D7S820 8 ▼ 12 ▼ 8,12	CSF1PO 9 ▼ 10 ▼ 9,10	D3S1358 14 ▼ 16 ▼ 14,16
TH01 6 ▼ 9 ▼ 6,9	D13S317 12 ▼ 12 ▼ 12,12	D16S539 11 ▼ 13 ▼ 11,13	D2S1338 18 ▼ 19 ▼ 18,19	D19S433 15 ▼ 16.2 ▼ 15,16.2
vWA 16 ▼ 17 ▼ 16,17	TPOX 8 ▼ 9 ▼ 8,9	D18S51 18 ▼ 21 ▼ 18,21	D5S818 11 ▼ 13 ▼ 11,13	FGA 21 ▼ 22 ▼ 21,22

At the bottom of the form, there is a "Deducible:" dropdown set to "No", a "DNA Template Amount (pg):" input field with the value "175", and three buttons: "Compare", "Reset", and "Back".

© 2016 City of New York Office of Chief Medical Examiner all rights reserved.

Figure C.3 - Comparison profile for "back strap and grips # 2"



Identifier

**Forensic Statistic Comparison Report**

v2.5

FB#1: FST report  
replicate

FB#2:

Item: swab of "front/back  
strap and grips # 2"Comparison: Kevin  
Johnson testDNA Template Amount (pg):  
175

Input By: admin

Hp: Kevin Johnson test (Comparison) + 2 Unknowns Hd: 3 Unknowns

Deducible: No

**Profiles**

Profile	D8S1179	D21S11	D7S820	CSF1PO	D3S1358	TH01	D13S317	D16S539	D2S1338	D19S433	vWA	TPOX	D18S51	D5S818	FGA
Kevin Johnson test (Comparison)															
Evidence															
1	12, 13, 14, 15, 16, 17	27, 28, 30, 31	10, 12	9, 10, 12	14, 15, 16, 17	6, 7, 9	11, 12	9, 11, 13	18, 19, 21	14, 15, 15.2	15, 16	6, 8	15	11, 12, 13	
2	13, 14, 15, 16, 17	27, 28, 30, 31	8, 12	12	14, 15, 16	6, 7, 9	11, 12	11, 12, 13	18, 19	11, 13, 14, 15, 15.2, 16.2	16, 17	8, 9	18, 21	11, 13	22
3															

**Comparison Result**

	Asian	Black	Caucasian	Hispanic
Likelihood Ratio	9.27e+08	1.39e+08	6.6e+07	2.04e+08

10/24/2016 4:36:54 PM

Figure C.4 - FST report reproducing results for "back strap and grips # 2"

636129238100946072-Caucasian.csv - Excel

FILEHOMEINSERTPAGE LAYOUTFORMULASDATAREVIEWVIEWDEVELOPERADD-INSTEAM

Figure C.5 - "Debug out" file for Caucasian subpopulation for "back strap and grips # 2" FST analysis

Forensic Statistical Tool v2.5

Home | Admin Welcome admin [LogOut](#)

Comparison + Unknown / 2 Unknowns Comparison + Unknown / 2 Unknowns (Edit)

FB#1:  Comparison:  FB#2:  Item:

**Evidence:**

Replicate 1 Replicate 2 Replicate 3

D8S1179 16 <input type="text"/> 13,16	D21S11 30 <input type="text"/> 27,28,29,30	D7S820 11 <input type="text"/> 8,10,11	CSF1PO 12 <input type="text"/> 10,12	D3S1358 16 <input type="text"/> 14,15,16
TH01 9 <input type="text"/> 6,7,9	D13S317 12 <input type="text"/> 11,12	D16S539 11 <input type="text"/> 9,11	D2S1338 18 <input type="text"/> 18	D19S433 16.2 <input type="text"/> 14,15,16.2
vWA 17 <input type="text"/> 16,17	TPOX 8 <input type="text"/> 8	D18S51 INC <input type="text"/>	D5S818 13 <input type="text"/> 13	FGA INC <input type="text"/>

**Kevin Johnson test (Comparison):**

Deducible:  No  DNA Template Amount (pg):

© 2016 City of New York Office of Chief Medical Examiner all rights reserved.

Figure C.6 - Replicate 1 for "slide grip grooves and mag release"

Forensic Statistical Tool v2.5

Home | Admin Welcome admin [LogOut](#)

Comparison + Unknown / 2 Unknowns Comparison + Unknown / 2 Unknowns (Edit)

FB#1:  Comparison:  FB#2:  Item:

**Evidence:**

Replicate 1 **Replicate 2** Replicate 3

D8S1179 16 <input type="text"/> 13,16	D21S11 28 <input type="text"/> 27,28	D7S820 11 <input type="text"/> 11	CSF1PO 12 <input type="text"/> 12	D3S1358 18 <input type="text"/> 14,16,18
TH01 9 <input type="text"/> 6,9	D13S317 12 <input type="text"/> 11,12	D16S539 13 <input type="text"/> 11,13	D2S1338 25 <input type="text"/> 18,19,25	D19S433 16.2 <input type="text"/> 14,15,16.2
vWA 17 <input type="text"/> 16,17	TPOX 8 <input type="text"/> 8	D18S51 18 <input type="text"/> 13,16,18	D5S818 13 <input type="text"/> 11,13	FGA INC <input type="text"/>

**Kevin Johnson test (Comparison):**

Deductible:  No  DNA Template Amount (pg):  119

© 2016 City of New York Office of Chief Medical Examiner all rights reserved.

Figure C.7 - Replicate 2 for "slide grip grooves and mag release"

The screenshot shows a web browser window with the URL `http://localhost:2926/frmManualEn` and a tab titled "FST - Comparison + Unkno...". The application is titled "Forensic Statistical Tool v2.5". The navigation bar includes "Home" and "Admin", with a welcome message "Welcome admin" and a "LogOut" link. The main content area is titled "Comparison + Unknown / 2 Unknowns" and "Comparison + Unknown / 2 Unknowns (Edit)".

At the top of the form, there are input fields for "FB#1:", "Comparison:", "FB#2:", and "Item:". Below these, there is a section titled "Evidence:" and a sub-section titled "Kevin Johnson test (Comparison):".

The "Kevin Johnson test (Comparison)" section contains a grid of 15 DNA marker comparison boxes, each with two dropdown menus and a text field for the comparison result:

D8S1179 16 ▼ 16 ▼ 16,16	D21S11 27 ▼ 30 ▼ 27,30	D7S820 8 ▼ 12 ▼ 8,12	CSF1PO 9 ▼ 10 ▼ 9,10	D3S1358 14 ▼ 16 ▼ 14,16
TH01 6 ▼ 9 ▼ 6,9	D13S317 12 ▼ 12 ▼ 12,12	D16S539 11 ▼ 13 ▼ 11,13	D2S1338 18 ▼ 19 ▼ 18,19	D19S433 15 ▼ 16.2 ▼ 15,16.2
vWA 16 ▼ 17 ▼ 16,17	TPOX 8 ▼ 9 ▼ 8,9	D18S51 18 ▼ 21 ▼ 18,21	D5S818 11 ▼ 13 ▼ 11,13	FGA 21 ▼ 22 ▼ 21,22

At the bottom of the form, there is a "Deducible:" dropdown set to "No", a "DNA Template Amount (pg):" input field with the value "119", and three buttons: "Compare", "Reset", and "Back".

© 2016 City of New York Office of Chief Medical Examiner all rights reserved.

Figure C.8 - Comparison profile for "slide grip grooves and mag release"

Identifiler

**Forensic Statistic Comparison Report**

v2.5

FB#1: FST report  
replicate

FB#2:

Item: swab of "slide grip  
grooves and mag  
release"Comparison: Kevin  
Johnson testDNA Template Amount (pg):  
119

Input By: admin

Hp: Kevin Johnson test (Comparison) + Unknown

Hd: 2 Unknowns

Deducible: No

**Profiles**

Profile	D8S1179	D21S11	D7S820	CSF1PO	D3S1358	TH01	D13S317	D16S539	D2S1338	D19S433	vWA	TPOX	D18S51	D5S818	FGA
Kevin Johnson test (Comparison)															
Evidence															
1	13, 16	27, 28, 29, 30	8, 10, 11	10, 12	14, 15, 16	6, 7, 9	11, 12	9, 11	18	14, 15, 16.2	16, 17	8		13	
2	13, 16	27, 28	11	12	14, 16, 18	6, 9	11, 12	11, 13	18, 19, 25	14, 15, 16.2	16, 17	8	13, 16, 18	11, 13	
3															

**Comparison Result**

	Asian	Black	Caucasian	Hispanic
Likelihood Ratio	2.04e+03	821.78	156.03	571.69

10/24/2016 7:05:12 PM

*Figure C.9 - FST reproducing results for "slide grip grooves and mag release"*

636129327114867377-Caucasian.csv - Excel

FILEHOMEINSERTPAGE LAYOUTFORMULASDATAREVIEWVIEWDEVELOPERADD-INSTEAM

Cut

Copy

Paste

Format Painter

Clipboard

Font

Calibri11

Alignment

Number

General

Conditional Formatting

Table

Normal

Bad

Good

Neutral

Insert

Delete

Format

Cells

Editing

AutoSum

Fill

Clear

Sort & Find & Filter

Select

A1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X			
1		Comparison + Known + Unknown / Known + 2 Unknowns																									
2		Deductible No																									
3		DNA Quar 119																									
4																											
5		Allele Frequencies				Numerator Dropout Rates				Drop-in Rates				Denominator Dropout Rates													
6		Allele	Frequency			Heterozygote		Homozygote						Heterozygote		Homozygote											
7		13	0.236			pHet0	0.93	pHom0	0.98	pC0	0.975			pHet0	0.93	pHom0	0.98										
8		11	0.281			pHet1	0.05	pHom1	0.02	pC1	0.02			pHet1	0.05	pHom1	0.02										
9		w	0.483			pHet2	0.02			pC2+	0.005			pHet2	0.02												
10		Comparison Profile				Evidence Profiles				Final Values																	
11		Known1	11,13			Rep 1		13		Num	1.23E-38																
12		Known2				Rep 2	11,13			Den	7.88E-41																
13		Known3				Rep 3																					
14		Known4								LR	156.0274																
15																											
16																											
17		Numerator																									
18		Gen1	Gen2	Gen3	Gen4	Freq1	Freq2	Freq3	Freq4	Rep1Drop	Rep1Drop	Rep1Drop	Rep1Drop	Rep1Cont	Rep2Drop	Rep2Drop	Rep2Drop	Rep2Drop	Rep2Cont	Rep3Drop	Rep3Drop	Rep3Drop	Rep3Drop	Rep3Cont	Product		
19		13,13	16,16			0.089685		1	1	0.98	0.98	0	0	0.975	0.98	0.98	0	0	0.975						0.07863		
20		13,16	16,16			0.019074		1	1	1	0.93	0.98	0	0	0.975	0.93	0.98	0	0	0.975					0.01506		
21		13,w	16,16			0.391884		1	1	1	0.05	0.98	0	0	0.975	0.05	0.98	0	0	0.975					0.00089		
22		16,16	16,16			0.002046		1	1	1	0.98	0.98	0	0	0.02	0.98	0.98	0	0	0.02					7.55E-0		
23		16,w	16,16			0.044748		1	1	1	0.05	0.98	0	0	0.02	0.05	0.98	0	0	0.02					4.30E-0		
24		w,w	16,16			0.466233		1	1	1	0.02	0.98	0	0	0.02	0.02	0.98	0	0	0.02					7.16E-0		
25		27,27	27,30			0.001058		1	1	1	0.98	0.93	0	0	0.005	0.98	0.05	0	0	0.02					4.72E-0		
26		27,28	27,30			0.005712		1	1	1	0.93	0.93	0	0	0.02	0.93	0.05	0	0	0.975					4.48E-0		
27		27,29	27,30			0.008316		1	1	1	0.93	0.93	0	0	0.02	0.05	0.05	0	0	0.02					7.19E-0		
28		27,30	27,30			0.011802		1	1	1	0.93	0.93	0	0	0.005	0.05	0.05	0	0	0.02					2.55E-0		
29		27,w	27,30			0.015288		1	1	1	0.05	0.93	0	0	0.005	0.05	0.05	0	0	0.02					1.78E-1		
30		28,28	27,30			0.022021		1	1	1	0.98	0.93	0	0	0.02	0.98	0.05	0	0	0.975					1.92E-0		
31		28,29	27,30			0.053856		1	1	1	0.93	0.93	0	0	0.975	0.05	0.05	0	0	0.975					0.00011		
32		28,30	27,30			0.076432		1	1	1	0.93	0.93	0	0	0.02	0.05	0.05	0	0	0.975					3.22E-0		
33		28,w	27,30			0.099008		1	1	1	0.05	0.93	0	0	0.02	0.05	0.05	0	0	0.975					2.24E-0		

636129327114867377-Caucasian

READY

Figure C.10 - "Debug out" for Caucasian subpopulation for "slide grip grooves and mag release" FST analysis

## Appendix D – Single-locus Testing

Forensic Statistical Tool v2.5

Home | Admin Welcome admin [LogOut](#)

Comparison + 2 Unknowns / 3 Unknowns Comparison + 2 Unknowns / 3 Unknowns (Edit)

FB#1:  Comparison: CSF single-locus test FB#2:  Item:

**Evidence:**

Replicate 1 Replicate 2 Replicate 3

D8S1179 INC <input type="text"/>	D21S11 INC <input type="text"/>	D7S820 INC <input type="text"/>	CSF1PO 14 <input type="text"/> 9,10,11,12,13,14	D3S1358 INC <input type="text"/>
TH01 INC <input type="text"/>	D13S317 INC <input type="text"/>	D16S539 INC <input type="text"/>	D2S1338 INC <input type="text"/>	D19S433 INC <input type="text"/>
vWA INC <input type="text"/>	TPOX INC <input type="text"/>	D18S51 INC <input type="text"/>	D5S818 INC <input type="text"/>	FGA INC <input type="text"/>

**CSF single-locus test (Comparison):**

Deducible: ☐ No ☐ DNA Template Amount (pg):  100

© 2016 City of New York Office of Chief Medical Examiner all rights reserved.

Figure D.1 - Replicate 1 for CSF testing where comparison alleles are observed in the replicate



Forensic Statistical Tool v2.5

Home | Admin Welcome admin [LogOut](#)

Comparison + 2 Unknowns / 3 Unknowns Comparison + 2 Unknowns / 3 Unknowns (Edit)

FB#1:  Comparison: CSF single-locus test FB#2:  Item:

**Evidence:**

Replicate 1 Replicate 2 Replicate 3

D8S1179 INC <input type="checkbox"/> <input type="text"/>	D21S11 INC <input type="checkbox"/> <input type="text"/>	D7S820 INC <input type="checkbox"/> <input type="text"/>	CSF1PO 14 <input type="checkbox"/> 9,10,11,12,13,14	D3S1358 INC <input type="checkbox"/> <input type="text"/>
TH01 INC <input type="checkbox"/> <input type="text"/>	D13S317 INC <input type="checkbox"/> <input type="text"/>	D16S539 INC <input type="checkbox"/> <input type="text"/>	D2S1338 INC <input type="checkbox"/> <input type="text"/>	D19S433 INC <input type="checkbox"/> <input type="text"/>
vWA INC <input type="checkbox"/> <input type="text"/>	TPOX INC <input type="checkbox"/> <input type="text"/>	D18S51 INC <input type="checkbox"/> <input type="text"/>	D5S818 INC <input type="checkbox"/> <input type="text"/>	FGA INC <input type="checkbox"/> <input type="text"/>

**CSF single-locus test (Comparison):**

Deductible: No ☐ DNA Template Amount (pg): 100  [Compare](#) [Reset](#) [Back](#)

© 2016 City of New York Office of Chief Medical Examiner all rights reserved.

Figure D.2 - Comparison for CSF testing where comparison alleles are observed in the replicate

Identifier

**Forensic Statistic Comparison Report**

v2.5

FB#1:                      FB#2:                      Item:                      Comparison: CSF single-locus test                      DNA Template Amount (pg): 100                      Input By: admin  
 Hp: CSF single-locus test (Comparison) + 2 Unknowns                      Hd: 3 Unknowns                      Deducible: No

**Profiles**

Profile	D8S1179	D21S11	D7S820	CSF1PO	D3S1358	TH01	D13S317	D16S539	D2S1338	D19S433	vWA	TPOX	D18S51	D5S818	FGA
CSF single-locus test (Comparison)				10,10											
Evidence															
1				9, 10, 11, 12, 13, 14											
2															
3															

**Comparison Result**

	Asian	Black	Caucasian	Hispanic
Likelihood Ratio	1	1	1	1

10/24/2016 7:14:28 PM

Figure D.3 - FST results for CSF testing where comparison alleles are observed in the replicate

636129332680115691-Asian.csv - Excel

FILEHOMEINSERTPAGE LAYOUTFORMULASDATAREVIEWVIEWDEVELOPERADD-INSTEAM

Cut

Copy

Paste

Format Painter

Clipboard

Calibri

11

A<sup>+</sup>

A<sup>-</sup>

Figure D.4 - "Debug out" data for Asian subpopulation for CSF testing where comparison alleles are observed in the replicate \*Note no other "Debug out" files were produced for this analysis

Forensic Statistical Tool v2.5

Home | Admin Welcome admin [LogOut](#)

Comparison + 2 Unknowns / 3 Unknowns Comparison + 2 Unknowns / 3 Unknowns (Edit)

FB#1:  Comparison: CSF single-locus test FB#2:  Item:

**Evidence:**

Replicate 1 Replicate 2 Replicate 3

D8S1179 INC <input type="text"/>	D21S11 INC <input type="text"/>	D7S820 INC <input type="text"/>	CSF1PO 14 <input type="text"/> 9,10,11,12,13,14	D3S1358 INC <input type="text"/>
TH01 INC <input type="text"/>	D13S317 INC <input type="text"/>	D16S539 INC <input type="text"/>	D2S1338 INC <input type="text"/>	D19S433 INC <input type="text"/>
vWA INC <input type="text"/>	TPOX INC <input type="text"/>	D18S51 INC <input type="text"/>	D5S818 INC <input type="text"/>	FGA INC <input type="text"/>

**CSF single-locus test (Comparison):**

Deducible: No  DNA Template Amount (pg): 100  [Compare](#) [Reset](#) [Back](#)

© 2016 City of New York Office of Chief Medical Examiner all rights reserved.

Figure D.5 - Replicate 1 for CSF testing where comparison alleles are not observed in the replicate

Forensic Statistical Tool v2.5

Home | Admin Welcome admin [LogOut](#)

Comparison + 2 Unknowns / 3 Unknowns Comparison + 2 Unknowns / 3 Unknowns (Edit)

FB#1:  Comparison: CSF single-locus test FB#2:  Item:

**Evidence:**

**CSF single-locus test (Comparison):**

D8S1179 INC <input type="checkbox"/> V <input type="checkbox"/>	D21S11 INC <input type="checkbox"/> V <input type="checkbox"/>	D7S820 INC <input type="checkbox"/> V <input type="checkbox"/>	CSF1PO 8 <input type="checkbox"/> 8 <input type="checkbox"/> 8,8 <input type="checkbox"/>	D3S1358 INC <input type="checkbox"/> V <input type="checkbox"/>
TH01 INC <input type="checkbox"/> V <input type="checkbox"/>	D13S317 INC <input type="checkbox"/> V <input type="checkbox"/>	D16S539 INC <input type="checkbox"/> V <input type="checkbox"/>	D2S1338 INC <input type="checkbox"/> V <input type="checkbox"/>	D19S433 INC <input type="checkbox"/> V <input type="checkbox"/>
vWA INC <input type="checkbox"/> V <input type="checkbox"/>	TPOX INC <input type="checkbox"/> V <input type="checkbox"/>	D18S51 INC <input type="checkbox"/> V <input type="checkbox"/>	D5S818 INC <input type="checkbox"/> V <input type="checkbox"/>	FGA INC <input type="checkbox"/> V <input type="checkbox"/>

Deductible: No ☐ DNA Template Amount (pg):

© 2016 City of New York Office of Chief Medical Examiner all rights reserved.

Figure D.6 - Comparison for CSF testing where comparison alleles are not observed in the replicate

Identifier

**Forensic Statistic Comparison Report**

v2.5

FB#1: FB#2: Item: Comparison: CSF single-locus test DNA Template Amount (pg): 100 Input By: admin  
 Hp: CSF single-locus test (Comparison) + 2 Unknowns Hd: 3 Unknowns Deducible: No

**Profiles**

Profile	D8S1179	D21S11	D7S820	CSF1PO	D3S1358	TH01	D13S317	D16S539	D2S1338	D19S433	vWA	TPOX	D18S51	D5S818	FGA
CSF single-locus test (Comparison)				8,8											
Evidence															
1				9, 10, 11, 12, 13, 14											
2															
3															

**Comparison Result**

	Asian	Black	Caucasian	Hispanic
Likelihood Ratio	1	1	1	1

10/24/2016 7:26:12 PM

Figure D.7 - FST results for CSF testing where comparison alleles are not observed in the replicate

636129339717868228-Asian.csv - Excel

FILEHOMEINSERTPAGE LAYOUTFORMULASDATAREVIEWVIEWDEVELOPERADD-INS

TEAM

Cut

Copy

Paste

Format Painter

Clipboard

Calibri

11

A<sup>+</sup>

A<sup>-</sup>

Figure D.8 - "Debug out" data for Asian subpopulation for CSF testing where comparison alleles are not observed in the replicate  
 \*Note no other "Debug out" files were produced for this analysis

## Curriculum Vitae

**Name:** Nathaniel D. Adams

**Current employer:** Forensic Bioinformatic Services, Inc.

**Address:** 2850 Presidential Drive, Suite 160  
Fairborn, OH 45324 USA

**Phone:** +1 (937) 426-9270 (office)

**Email:** [Adams.201@Wright.edu](mailto:Adams.201@Wright.edu)  
[Adams@Bioforensics.com](mailto:Adams@Bioforensics.com)

**Present position:** Systems Engineer

**Current duties:** Operating custom and commercial software for independent review of forensic STR DNA analysis; reviewing materials generated during the course of STR DNA testing; performing data analysis for research and case work; and reviewing current scientific and industry literature.

**Educational background:** **B.S.** Computer Science  
Wright State University  
Fairborn, OH

**M.S.** Computer Science, in progress, 2014-present  
Wright State University  
Fairborn, OH

**Professional experience:** Systems Engineer  
Forensic Bioinformatic Services, Inc.  
Fairborn, OH  
2012 – present

**Presentations:** N. Adams. The Science and the State of Probabilistic Genotyping. Questioning Forensics: Inside the Black Box. The Legal Aid Society. October 28, 2016.

N. Adams. Weka in Java: Classification and Clustering. Data Science and Security Cluster, Wright State University. September 2, 2016. Fairborn, OH.



N. Adams, D. Krane. Black Boxes and Due Process: Transparency in Expert Software Systems. 68th Annual Meeting of the American Academy of Forensic Sciences. February 26, 2016. Las Vegas, Nevada.

D. Krane, C. Rowland, N. Adams. Disputed DNA Stats for a Low-level Sample: A Case Study. 68th Annual Meeting of the American Academy of Forensic Sciences. February 26, 2016. Las Vegas, Nevada.

N. Adams, R. Chakraborty, C. Rowland, D. Krane. Complex Mixtures and the Minimum Number of Contributors: A Case Study (poster). 68th Annual Meeting of the American Academy of Forensic Sciences. February 26, 2016. Las Vegas, Nevada.

R. Koppl, D. Krane, N. Adams. Minimizing and Leveraging Bias in Forensic Science. National Institute of Standards and Technology International Symposium on Forensic Science Error Management. July 24, 2015. Washington, DC.

N. Adams. Is the Tail Wagging the Dog: Reviewing Probabilistic Genotyping Software. National Forensic College of National Academy of Criminal Defense Lawyers and Cardozo Law School. June 10, 2015. New York City, NY.

N. Adams. Introduction to Pattern Mining. Bioinformatics Research Group, Wright State University. March 24, 2015. Fairborn, OH.

S. Al-Awadi, M. Sabbaha, N. Adams, A. Marshall, C. Rowland, D. Krane. Pairwise Comparisons as a Means of Validating Iraqi Muslim and Christian Allele Frequency Databases (poster). 67th Annual Meeting of the American Academy of Forensic Sciences. February 19, 2015. Orlando, FL.

N. Adams, A. Marshall. The Science (and Pseudoscience) of DNA Profiling. The Kettering-Moraine Public Library. June 21, 2014. Kettering, OH.

N. Adams. Allele Frequencies: How the Same Statistic Varies and Why It Should. Forensic DNA for Trial Attorneys, Cook County Office of the Public Defender. May 30, 2014. Chicago, IL.

D. Krane, N. Adams. The Science (and Pseudoscience) of DNA Profiling. Wright State University Foundation Board of Trustees Meeting, Wright State University. October 24, 2013. Fairborn, OH.

N. Adams, A. Marshall. Forensic DNA Analysis. Bioinformatics Research Group, Wright State University. April 3, 2013. Fairborn, OH.

**Continuing education:** Questioning Forensics: Inside the Black Box. The Legal Aid Society. October, 2016. New York, NY.

DNA and More: Developments in Forensic Science, Cook County Office of the Public Defender. June, 2016. Chicago, IL.

68th Annual Meeting of the American Academy of Forensic Sciences. February, 2016. Las Vegas, NV.

National Institute of Standards and Technology International Symposium on Forensic Science Error Management. July, 2015. Washington, DC.

National Forensic College of the National Academy of Criminal Defense Lawyers and Cardozo Law School. June, 2015. New York City, NY.

Forensic Bias and Error: Causes and Correction, Cook County Office of the Public Defender. May, 2015. Chicago, IL.

DNA Mixture Interpretation Software Workshop, Midwestern Association of Forensic Scientists. June, 2014. St. Louis, MO.

67th Annual Meeting of the American Academy of Forensic Sciences. February, 2015. Orlando, FL.

Science in the Courtroom for the 21<sup>st</sup> Century: Current Issues in DNA Litigation. Cook County Office of the Public Defender. May, 2013. Chicago, IL.

**Professional organizations:** Institute of Electrical and Electronics Engineers. Student member.

Association for Computing Machinery. Student member.

**Testimony:** State v. Emanuel Fair. King County Superior Court, Seattle, WA. Pretrial hearing on probabilistic genotyping.

**Student organizations and recognition:** Data Science and Security Cluster, 2015-present. Wright State University, Fairborn, OH.

Bioinformatics Research Group, 2012-present. Wright State University, Fairborn, OH.

Dean's Leadership Institute, 2013-2014. College of Engineering and Computer Science. Wright State University, Fairborn, OH.

College Award for Senior Design, College of Engineering and Computer Science, May, 2014. Wright State University, Fairborn, OH.

Choose Ohio First Scholarship

Recognition for Excellence in Service-Learning, 2013. Wright State University, Fairborn, OH.

Lockheed Martin Scholarship